Mini-batch gradient descent

→ if mini-batch size = m ⇒ Batch Gradient Descent $(X^{\{1\}}, Y^{\{1\}}) = (X, Y)$

" = 1 ⇒ Stocastic Gradient Descent.

Every example is a mini-batch. $(X^{\{1\}}, Y^{\{1\}}) = (X^{(1)}, Y^{(1)})$

In practice, m in mini-batch gradient descent is some what in between 1 and m

Stochastic gradient descent	In-between (mini-batch size not too big or small)	Batch gradient descent (mini-batch size=m)
↓	↓	↓
lose speedup from Vectorization.	Fastestest Learning • Vectorization (~1000) • Make progress without processing entire training set.	Too long per iteration

Mini-Batch gradient descent: Repeat n epochs

for $t = 1, \dots, 5000$: mini-batch iterations.

forward prop on $X^{\{t\}}$

Compute cost $J^{\{t\}} = \frac{1}{1000} \sum_{i=1}^{1000} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \cdot 1000} \sum_{l} \| W^{[l]} \|_F^2$

backprop to compute gradients

$W^{[l]} := W^{[l]} - \alpha dW$, $b^{[l]} := b^{[l]} - \alpha db$

Choosing mini-batch size:

If small training set → use batch gradient descent
($m < 2000$).

Typical mini-batch size

→ 64, 128, 256, 512

2^6 2^7 2^8 2^9

Theoretically 2^n makes gradient descent run faster.